QASPARSE: QUERY-ADAPTIVE SPARSE ATTENTION WITH TAILORED BUDGETS AND TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-sequence inference in large language models is constrained by the high cost of prefilling, where dense attention scales quadratically with context length. Exploiting similarity across attention heads offers a way to reduce redundant computation. Prior approaches in this direction have relied on localized query observations to determine head-level computation reuse or budget allocation. However, inter-head similarity does not imply uniform query requirements, leaving adaptation to varying budget needs and selection preferences across query regions a key challenge. We introduce ***QASparse****, a* ***Q****uery-****A****daptive* ***Sparse*** *attention mechanism* with hardware-efficient execution. QASparse preserves the efficiency of shared cross-head importance estimation while adapting computational budgets and token selections to query requirements, and it executes these dynamic sparse decisions through dedicated kernels. Its query adaptivity stems from two complementary innovations: Local Budget Regression models local variations in computational demand along query positions to estimate budgets for different query regions and attention heads; Boundary Voting lets queries within each head vote over candidates near the selection cutoff using actual query/key scores, refining the retained set while preserving the initial selection size. To reduce the execution overhead of dynamic budgets, we further design length-specialized selection kernels that efficiently support query-adaptive sparse patterns across context lengths. Together, these designs efficiently combine shared estimation with local decisions while adapting both how much attention is computed and which interactions are retained. Experimental results demonstrate improved benchmark scores and faster prefill attention computation compared with existing sparse attention baselines.

1 INTRODUCTION

Large language models (LLMs) are applied to tasks such as long-document understanding and repository-level code processing, motivating efficient long-sequence inference (Liu et al., 2025; Huo et al., 2026). However, as input sequences grow longer, dense attention in the prefill stage performs QK computation for all causally visible tokens, incurring quadratic computational cost in sequence length and potentially becoming a major bottleneck in long-context prefilling (Zhao et al., 2026a; Fu, 2024). Efficient dense-attention implementations improve execution efficiency through tiling, fusion, and parallelization (Dao, 2024; You et al., 2026), while retaining QK computation that grows quadratically with input length. To reduce this cost, sparse attention performs QK computation only for tokens deemed important, omitting the remaining QK computation (Lai et al., 2025; Fan et al., 2026b; Zhang et al., 2026; Jones et al., 2026; Jo et al., 2026; Song et al., 2026). Effective sparse acceleration therefore requires selecting important tokens with low overhead while largely preserving dense-attention capabilities.

To reduce the importance-estimation overhead of constructing sparse indices, a class of methods compresses queries and keys along the sequence dimension and constructs sparse indices from correlations among the compressed representations (Lai et al., 2025; Wang et al., 2026a; Liu et al., 2026a;b). Although this approach reduces the number of representations involved in estimation, it may obscure fine-grained correlations and distort block-importance estimates. Another line of work exploits similarities across attention heads to reduce redundant computation by sharing attention scores or sparse patterns (Wang et al., 2026b; Agarwal et al., 2024; Peng et al., 2025). Proxy-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Attn (Wang et al., 2026b) analyzes the similarity among different attention heads and uses pooled representative heads to provide shared importance rankings, together with online head-level budget estimation for sparse index construction. However, relying on a single shared ranking and budgets derived from local Q observations cannot accommodate the varying budget demands across query positions or the head-specific token preferences of individual queries. Consequently, queries may receive excessive or insufficient budgets and fail to select the most relevant tokens, due to the lack of query adaptivity.

To this end, we propose ***QASparse**, a **Q**uery-**A**daptive **Sparse** attention mechanism* with hardware-efficient execution, which adaptively allocates computational budgets and selects KV tokens for different query regions and attention heads. QASparse achieves this query adaptivity through two complementary mechanisms: (1) Local Budget Regression uses attention statistics from sampled queries to model local budget variation, estimating suitable budgets across query regions and attention heads; (2) Boundary Voting uses the predicted budget to locate the selection boundary, then computes QK scores for candidates near that boundary to finalize the retained set. Together, these mechanisms enable QASparse to assign more suitable computational budgets to different query regions while maintaining low selection overhead by restricting additional decisions to a small number of boundary candidates. Furthermore, to reduce the execution overhead of dynamic token-selection decisions, we design Length-Specialized Kernels with dedicated execution paths tailored to different context lengths, thereby efficiently supporting query-adaptive sparse patterns across sequence lengths.

We evaluate QASparse on Llama3.1-8B-Instruct (Grattafiori et al., 2024), MegaBeam-Mistral-7B-512k (Wu & Song, 2025), GLM-4-9B-Chat-1M (Team GLM et al., 2024), and Qwen2.5-7B-Instruct-1M (Yang et al., 2025), using RULER (Hsieh et al., 2024) and InfiniteBench (Zhang et al., 2024a) to assess performance across controlled context lengths and long-range dependencies in very long contexts, respectively. Experimental results show that, compared with existing sparse-attention baselines, QASparse improves benchmark performance while accelerating attention computation during the prefill stage. Our method achieves a 2.81× speedup over FlashAttention-2 at 128K context length.

2 Sparse Attention: Observations and Motivation

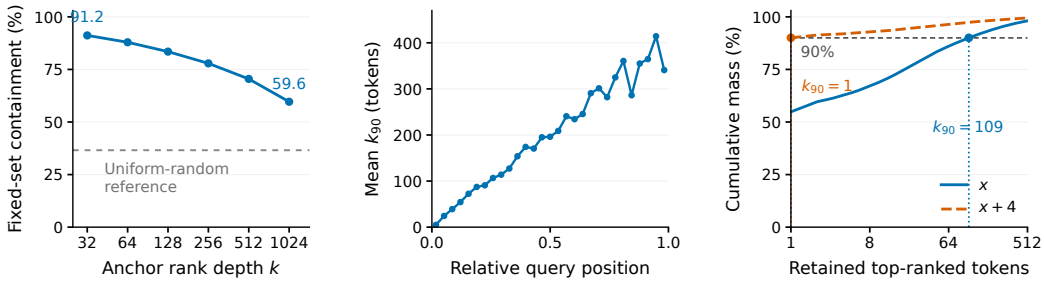
Sparse Attention Balances Fidelity and Efficiency Sparse attention reduces computational cost by omitting substantial QK computation while preserving dense-attention performance. For query q in head h, let $S_{h,q}$ denote the selected key tokens, and let $x_{h,q}$ and $y_{h,q}(S_{h,q})$ denote the dense and sparse attention outputs, respectively. Sparse selection can be formulated as

$$\min_{S} \sum_{h,q} \left[\|x_{h,q} - y_{h,q}(S_{h,q})\|_2^2 + \lambda |S_{h,q}| \right], \tag{1}$$

which balances output approximation against QK computation. However, QK computation alone does not determine latency, which also depends on GPU parallelism and workload balance. Query-Adaptive sparse attention must therefore jointly optimize algorithmic and execution efficiency while preserving selection semantics.

Attention Heads Share Salient Context Positions In LLM attention, highly ranked tokens in one head often overlap with important tokens in other heads, although this overlap weakens as the rank range expands (Figure 1(a)). This cross-head sharing enables shared ranking, where pooled Q/K representations provide a coarse ordering reusable across heads at low cost (Wang et al., 2026b). However, shared importance does not imply identical relative importance across heads, so token selection must also account for each head's attention distribution.

Varying Queries Require Adaptive Attention Budgets Prior work (Wang et al., 2026b) estimates a head-level budget from the final query block and reuses it along the query axis. However, the budget required for the same attention coverage varies with query position, showing both global trends and local fluctuations (Figure 1(b,c)). A fixed head-level budget may therefore over- or under-allocate tokens, motivating local budget estimation along the query axis.



(a) Cross-head sharing (b) Budget variation (c) Local budget drop

Figure 1: **(a)** Cross-head Top-k overlap decreases as k increases. **(b)** The budget for target attention coverage grows overall with query position but exhibits local variation. **(c)** Nearby queries can require substantially different budgets for the same 90% coverage.

These observations motivate reusing cross-head importance estimates while adapting budgets locally along the query axis and refining candidate selection with head-specific evidence. The next section develops Local Budget Regression and Boundary Voting around these two forms of adaptation.

3 Query-Adaptive Selection

Our method reuses a shared ranking method (Wang et al., 2026b): We average-pool Q and K vectors across heads at matching positions and compute their attention scores to obtain a shared token ranking. Building on this shared ranking, we propose **Query-Adaptive Selection**, which consists of three modules: (1) **Local Budget Regression**, (2) **Boundary Voting**, and (3) **Length-Specialized Kernels**.

3.1 Local Budget Regression

The budget variation observed in Section 2 (Figures 1b and 1c) motivates local estimation along the query axis: budget demand can increase overall while exhibiting local dips, and nearby queries can have substantially different requirements.

Representative Query Sampling. We select a representative query $\widehat{q}$ every s positions along the query sequence and compute its attention distribution $\widehat{a}$, normalized over all causally visible key tokens:

$$\widehat{a} = \mathrm{softmax}\left(\frac{K\widehat{q}}{\sqrt{d}}\right). \tag{2}$$

Budget Estimation along the Shared Ranking. We accumulate the target head's attention weights over visible tokens along the shared ranking π. The smallest number of tokens required to reach the target coverage γ gives the budget observation b:

$$b = \min\left\{ m : \sum_{j=1}^{m} \widehat{a}_{\pi(j)} \geq \gamma \right\}. \tag{3}$$

Although the ordering is shared across heads, the attention weights are head-specific, allowing the same ranking to yield different budgets.

Comparing Regression Schemes. Figure 1c in Section 2 shows a sharp budget drop between nearby queries. Directly assigning sampled budgets to neighboring queries may propagate the influence of outlying observations to local allocations. We therefore fit linear models to budget trends within neighboring regions and consider three schemes.

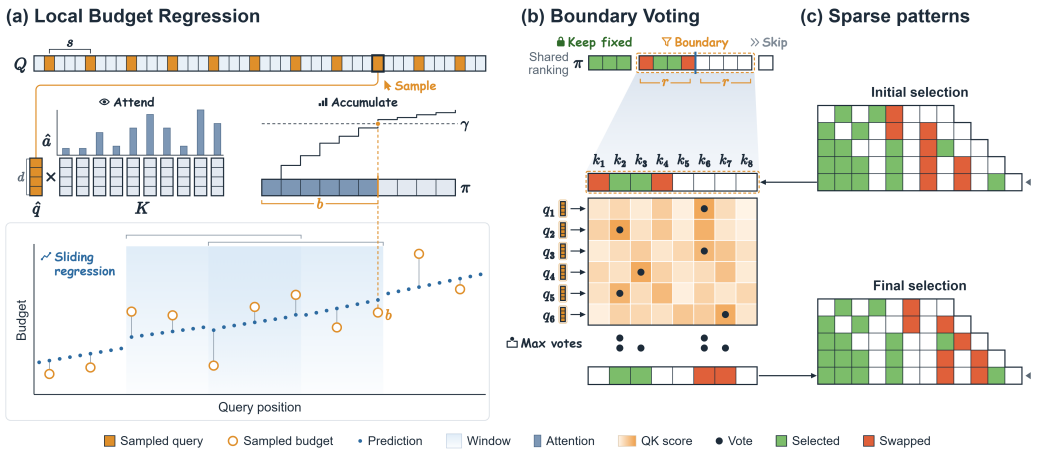


Figure 2: Overview of Query-Adaptive Selection. **(a)** Budgets are estimated from regularly sampled queries, and sliding-window regression predicts token requirements. **(b)** QK scores are used to vote among tokens near the cutoff of the shared ranking, refining the selection while preserving the budget. **(c)** Sparse attention patterns before and after Boundary Voting.

- **Global Linear Regression.** The overall increase in budget with query position in Figure 1b motivates fitting one line to all sampled budgets. However, a single slope cannot also capture the local drops shown in Figure 1c, so we do not adopt this scheme.
- **Non-overlapping Piecewise Linear Regression.** We divide sampled positions into disjoint consecutive segments and fit each segment independently. Different regions can have different slopes, but predictions draw on a limited range of observations. In particular, predictions at internal segment boundaries cannot incorporate information from neighboring segments over a broader range. We therefore do not adopt this scheme.
- **Sliding-window Linear Regression.** We fit overlapping consecutive windows and average the predictions from all windows covering each position. Interior positions can thus use neighborhoods that cross fixed segment boundaries, reducing dependence on a single partition. We adopt this scheme to model local budget variation and aggregate overlapping predictions to use observations from a broader neighborhood when estimating the budget at each query position.

Query-adaptive Budget Assignment. We assign token budgets to query positions based on the regression predictions and obtain an initial selection from the shared ranking. Boundary Voting then refines its membership while preserving the number of selected tokens. This adaptive estimation assigns different budgets along the query axis of each head and uses neighborhood information to capture local demand. Sampled-query coverage guides allocation, balancing local adaptation with estimation cost.

3.2 Boundary Voting

The cross-head sharing observed in Section 2 (Figure 1a) motivates reusing an importance ranking, but shared attention to a position does not imply identical relative importance across heads. Once the budget is determined, we therefore refine candidate selection with evidence from the target head. Coarse-grained compression can miss a few high-attention positions and impair model performance. We therefore prioritize these interactions, with the decoding experiment in Appendix A providing further support for this selection principle.

Boundary Candidate Construction. We take r tokens on each side of the cutoff in the shared ranking to form the boundary candidate set C. The higher-ranked token prefix outside this range remains unchanged.

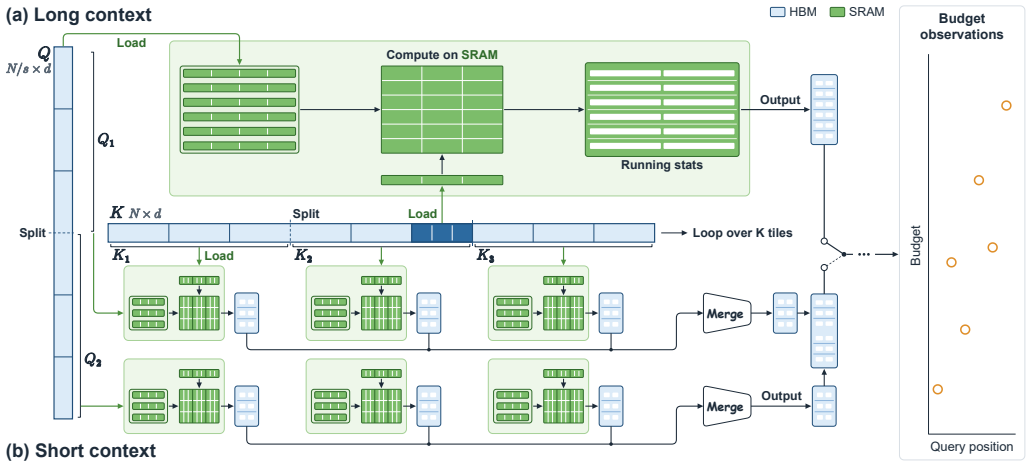


Figure 3: **(a) Long context:** The kernel keeps a larger, intact group of sampled Q vectors in SRAM and streams through all visible K blocks within one logical workload. **(b) Short context:** The kernel splits the same Q workload into smaller groups Q_1 and Q_2 and pairs each group with the K partitions.

Head-specific Max Voting. We locally sample n query vectors to form the set T, and each query casts one vote for the candidate token with the highest QK score. With k_u denoting the key of token u, the vote count for token j is

$$v_j = \sum_{q \in T} \mathbf{1}\left[j = \arg\max_{u \in C} q^\top k_u \right]. \tag{4}$$

Only causally visible pairs are compared. Each vote reflects a query's strongest attention preference, while the accumulated votes combine preferences across local queries. To reduce the QK computation required for voting, we subsample queries and keys with stride t along their respective sequence positions and approximate the voting procedure above using only the matching scores between the sampled vectors.

Budget-preserving Selection. We rerank the boundary tokens according to the vote count v_j received by each key vector, retain the same number of tokens as in the initial selection within this range, and combine them with the fixed prefix to obtain the sparse index set. This procedure combines a shared ordering with head-specific attention information to refine which candidates are retained under a given budget. Concentrating additional computation near the cutoff balances the efficiency of shared estimation with the flexibility of candidate selection.

3.3 Length-Specialized Kernels

Prior work (Wang et al., 2026b; Lai et al., 2025) often reuses a single attention kernel across context lengths. However, workload size and the amount of schedulable parallel work vary with context length across stages. We therefore design dedicated selection kernels for different context lengths to reduce token-selection overhead. We use **Budget Estimation along the Shared Ranking**, which computes budgets from sampled queries along the shared ranking, as a representative path.

Length-specialized workload granularity. For long contexts, the Budget Estimation in the Shared Ranking stage can load the working sets of multiple sampled queries into fast on-chip storage, keep them resident, and stream visible K vectors to compute against these resident queries.For short contexts, however, an oversized query working set can substantially reduce the number of resulting work units. To increase the number of work units and improve parallelism, we split the sampled-query work that would otherwise be processed together into smaller sampled-query sets and assign them to more independent execution resources.

Context-dependent QK decomposition. For long contexts, a query group can traverse a continuous visible key range and complete its computation against all visible key vectors as one logical workload. For short contexts, the K range associated with a query group is divided into multiple key-range subproblems so that the work can be distributed across more execution resources and scheduled independently.

The long-context path uses a reuse-oriented organization to reduce repeated Q/K preparation, whereas the short-context path partitions both query work and K ranges to enlarge the independently schedulable computation. Length-Specialized Kernels thereby improve workload distribution across execution resources and increase GPU parallelism.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

In this section, we provide a comprehensive overview of the models and datasets employed in our experiments to assess the performance and capabilities of various long-context language models.

Models We use four long-context instruction-following or chat models: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct-1M (Yang et al., 2025), MegaBeam-Mistral-7B-512k (Wu & Song, 2025), and GLM-4-9B-Chat-1M (Team GLM et al., 2024). MegaBeam-Mistral-7B-512k was further trained from Mistral-7B-Instruct-v0.2. These models span different model families and long-context configurations, allowing us to examine the applicability of the proposed method across architectures and context settings.

Datasets We evaluate the models on two benchmarks with different long-context challenges. (i) RULER (Hsieh et al., 2024) a synthetic benchmark dataset created to evaluate long-context LLMs with customizable sequence lengths and task complexities. It extends the basic needle-in-a-haystack test as well as introduces new task categories such as multi-hop tracing and aggregation. (ii) InfiniteBench (Zhang et al., 2024a) a benchmark dataset designed to test LLMs' understanding of long dependencies within extensive contexts, with an average token count of 214k. It includes 10 synthetic and real-world tasks across various domains.

Implementation Details All experiments are conducted on a single NVIDIA A100 64GB GPU. We implement a custom inference pipeline in PyTorch (Paszke et al., 2019), build on FlashAttention (Dao, 2024) for efficient long-context attention computation, and use Triton (Tillet et al., 2019) to optimize GPU computation. We set block_size $= 128$ in all experiments. We set the cumulative attention threshold to $\gamma = 0.95$ for all models and allocate computation budgets adaptively. For Qwen2.5-7B-Instruct-1M, we additionally impose a minimum budget of 2048 tokens.In Local Budget Regression, we sample one representative query at the midpoint of each query block, with an interval of $s = 128$ query positions. Each sliding regression window contains $w = 10$ consecutive sampled queries and advances by two sampled queries.In Boundary Voting, we take at most eight candidate key blocks on each side of the shared-ranking cutoff. Voting is organized in 128×128 QK blocks, and both query and key vectors are subsampled with stride $t = 8$.In the final attention mask, the last two query blocks attend to all causally visible key blocks. For all heads in each layer, we use a shared ranking to guide token selection across all heads.All experiments use greedy decoding to maintain consistency across results.

Baselines We compare QASparse with three baselines covering efficient dense attention and adaptive sparse attention:(1) **FlashAttention** (Dao, 2024) is an optimized attention implementation that uses hardware-aware computation and efficient memory access to reduce attention computation and memory overhead. We use it as the exact dense-attention baseline for comparing computational savings and model performance.(2) **FlexPrefill** (Lai et al., 2025) dynamically selects between query-aware and vertical-slash sparse patterns according to the input and attention head. We set its pattern-selection threshold to $\tau = 0.1$ and its cumulative attention threshold for sparse index selection to $\gamma = 0.95$.(3) **ProxyAttn** (Wang et al., 2026b) combines cross-head shared importance rankings with independently estimated head budgets. We configure one proxy head per layer and each head then

estimates its own budget with $\gamma = 0.95$ and applies that budget to all its query positions, reusing ranking computation while retaining head-specific budgets.

All sparse baselines perform sparse computation during prefill and switch to dense computation during decoding.

4.2 Main Results

This section presents a comparative analysis of various attention mechanisms. The evaluation includes FlashAttention, FlexPrefill, ProxyAttn, and our proposed method. The comparison aims to demonstrate our approach's efficiency and effectiveness against existing methods.

RULER. QASparse achieves the highest mean score among sparse methods on all four models and outperforms both sparse baselines at 128k (Table 1).

Table 1: Performance comparison of different methods on various models and sequence lengths on RULER. The best results are highlighted in bold.

Models	Methods	4k	8k	16k	32k	64k	128k	Avg
Llama	FlashAttention	95.18	93.36	93.26	89.28	85.56	77.96	89.10
	FlexPrefill	94.59	**93.53**	92.58	91.59	85.18	73.67	88.52
	ProxyAttn	94.12	92.25	93.35	90.34	87.24	75.15	88.74
	QASparse	**94.90**	93.21	**93.75**	**92.22**	**87.70**	**77.82**	**89.93**
GLM	FlashAttention	94.26	92.68	91.42	89.27	85.04	84.45	89.52
	FlexPrefill	92.46	91.40	90.64	87.69	86.70	84.34	88.87
	ProxyAttn	95.00	**93.67**	91.73	**90.57**	**87.76**	85.45	90.70
	QASparse	**95.05**	93.46	**92.97**	90.39	87.53	**86.03**	**90.91**
Mistral	FlashAttention	93.34	90.60	86.69	86.72	81.37	83.75	87.08
	FlexPrefill	90.05	88.81	86.00	84.20	81.56	82.62	85.54
	ProxyAttn	93.51	90.84	**88.43**	**87.09**	**83.80**	84.39	88.01
	QASparse	**94.21**	**92.30**	87.96	86.94	83.00	**85.30**	**88.29**
Qwen	FlashAttention	94.81	91.75	93.47	90.15	87.46	82.80	90.07
	FlexPrefill	93.35	91.06	91.94	87.03	85.84	80.14	88.23
	ProxyAttn	94.17	91.91	92.42	87.60	86.04	81.17	88.88
	QASparse	**94.35**	**92.21**	**92.91**	**88.95**	**86.54**	**82.52**	**89.58**

InfiniteBench. QASparse leads the sparse baselines in mean score on all four models, with strong long-book QA, summarization, and retrieval results (Table 2).

Table 2: Performance comparison of different methods on various models and tasks on Infinite Bench. The bold numbers indicate the first highest value in each column.

Models	Methods	En.Sum	En.QA	En.MC	En.Dia	Zh.QA	Code.Run	Math.Find	Retr.Passkey	Retr.Number	Retr.KV	Avg
Llama	FlashAttention	35.22	34.98	81.82	20.00	42.67	0.00	16.67	96.67	100.00	60.00	48.80
	FlexPrefill	32.87	28.01	72.73	**30.00**	36.40	0.00	22.22	96.67	100.00	48.00	46.69
	ProxyAttn	32.09	33.22	72.73	10.00	**43.72**	0.00	22.22	93.33	100.00	40.00	44.73
	QASparse	**35.13**	**38.58**	**72.73**	10.00	41.42	**0.00**	**22.22**	**96.67**	**100.00**	**56.00**	**47.28**
GLM	FlashAttention	29.32	28.57	72.73	30.00	17.10	0.00	27.78	96.67	90.00	0.00	39.22
	FlexPrefill	**34.92**	28.08	72.73	20.00	16.01	0.00	16.67	**100.00**	**83.33**	0.00	37.17
	ProxyAttn	31.13	27.49	72.73	20.00	**17.62**	0.00	**27.78**	96.67	80.00	0.00	37.34
	QASparse	28.78	**28.47**	**72.73**	**30.00**	17.52	**0.00**	22.22	96.67	80.00	**0.00**	**37.64**
Mistral	FlashAttention	26.74	11.13	54.55	0.00	19.45	0.00	27.78	100.00	100.00	68.00	40.77
	FlexPrefill	24.78	11.05	63.64	0.00	10.17	0.00	27.78	100.00	100.00	76.00	41.34
	ProxyAttn	**26.04**	10.47	54.55	0.00	14.47	0.00	**33.33**	100.00	100.00	64.00	40.29
	QASparse	24.89	**14.19**	**63.64**	**0.00**	**16.54**	**0.00**	27.78	**100.00**	**100.00**	**76.00**	**42.30**
Qwen	FlashAttention	34.68	9.66	90.91	30.00	11.32	0.00	44.44	30.00	13.33	0.00	26.43
	FlexPrefill	31.35	**9.77**	90.91	20.00	10.94	0.00	55.56	**26.67**	10.00	0.00	25.52
	ProxyAttn	31.28	8.69	81.82	20.00	13.45	0.00	44.44	10.00	6.67	0.00	21.64
	QASparse	**38.32**	8.08	**90.91**	**20.00**	**13.85**	**0.00**	**55.56**	23.33	**13.33**	**0.00**	**26.34**

Performance vs. Latency. Tuning γ balances accuracy and latency. Figure 4 shows improved trade-offs on average and at 4k and 64k; Appendix I reports the full measurements.

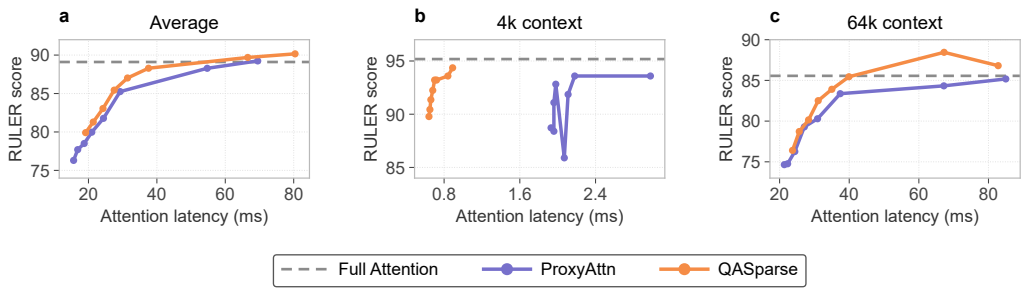


Figure 4: RULER quality–latency trade-offs. At 4k, QASparse's specialized short-context kernel supports approximately 3× speedup over ProxyAttn at comparable accuracy. The dashed line denotes Full Attention.

4.3 Ablations

We examine the effects of kernel specialization, budget regression, shared-ranking count, and Boundary Voting candidate-block count. Appendices B–E report the full results. Additional ablations of full attention on the final two query blocks, budget-label ordering, and voting scores are described in Appendices F–H.

Length-Specialized Kernels. We compare the short-context and long-context selection kernels at seven lengths from 8k to 512k. The short-context kernel is faster at 8k and 16k, whereas the long-context kernel is faster at every tested length from 32k onward (Figure 5). This crossover supports adapting the work decomposition to context length: finer partitions favor short contexts, while larger work units with greater reuse favor long contexts. Appendix D reports the selection-kernel latencies.

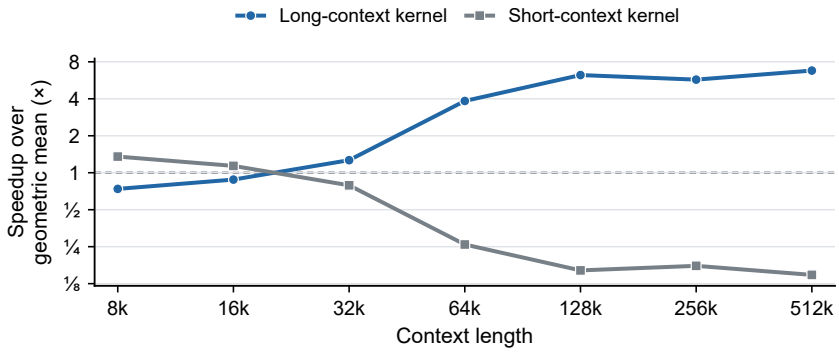


Figure 5: The horizontal 1× line marks the geometric-mean latency of the two kernels at each context length. Speedup is this reference latency divided by the latency of the indicated kernel.

Budget regression. We compare **Global Linear Regression**, **Non-overlapping Piecewise Linear Regression**, and **Sliding-window Linear Regression**, as defined in Section 3. We keep representative-query sampling and budget-label computation unchanged and vary the regression scheme. On Llama-3.1-8B-Instruct, both local schemes achieve higher mean RULER scores than the global fit, supporting local estimation of query-dependent budgets. Sliding-window Linear Regression achieves the highest mean RULER score (Appendix E).

Number of shared rankings. We compare ProxyAttn and QASparse with 1, 2, and 4 shared rankings on GLM and Qwen, and with 1, 2, 4, and 8 on Llama and MegaBeam. With one shared ranking, QASparse achieves a higher mean RULER score than ProxyAttn on all four models. Reducing the count from the largest tested value to one also preserves accuracy better for QASparse on each model. QASparse therefore tolerates greater ranking sharing in these experiments, supporting the use of one shared ranking per layer. Appendix B provides the per-length scores.

Boundary Voting candidate blocks. We vary the total number of candidate key blocks around the shared-ranking cutoff among 4, 8, 16, and 32, with Boundary Voting preserving the selected-block budget in each setting. The default 16-block region uses up to eight blocks on each side of the cutoff. On Llama-3.1-8B-Instruct, this setting achieves the highest mean RULER score, supporting the default candidate region (Appendix C).

5 Related Work

Long Context LLMs. Existing research broadly follows three routes: continued pretraining or parameter-efficient fine-tuning on long text to adapt models to longer positional distributions(Fu et al., 2024; Chen et al., 2024; Xiong et al., 2024; Yang et al., 2025; DeepSeek-AI et al., 2026); modifying positional representations or context extrapolation schemes to enlarge the window without fully retraining the model(Peng et al., 2024; Ding et al., 2024; Shang et al., 2025; Gelberg et al.,

2026; Zhang et al., 2024b); and introducing external memory, retrieval, or compression so that every decoding step need not attend to all historical tokens with equal intensity(Mohtashami & Jaggi, 2023; Tworkowski et al., 2023; Xu et al., 2024; Munkhdalai et al., 2024; Jiang et al., 2024b). Longformer(Beltagy et al., 2020) and BigBird(Zaheer et al., 2020) establish fixed sparse architectures based on local, global, and random connections.

LLM Inference Acceleration. Inference acceleration can be organized around exact-attention kernels, model-level sparsity, and KV-cache access. FlashAttention(Dao et al., 2022), FlashAttention-2(Dao, 2024), and FlashInfer(Ye et al., 2025) optimize exact attention through IO-aware execution, parallel work partitioning, and serving-oriented kernels. Native Sparse Attention(Yuan et al., 2025), DeepSeek-V3.2(DeepSeek-AI et al., 2025), and DeepSeek-V4(DeepSeek-AI et al., 2026) incorporate executable sparsity into training or model design. Prefill methods such as MInference(Jiang et al., 2024a), FlexPrefill(Lai et al., 2025), QuoKA(Jones et al., 2026), VSPrefill(Chen, 2026), and FlashPrefill V2(Fan et al., 2026a) select context-dependent token subsets, whereas decode methods reduce or selectively access the KV cache through H^2O(Zhang et al., 2023), StreamingLLM(Xiao et al., 2024), SnapKV(Li et al., 2024), Quest(Tang et al., 2024), DuoAttention(Xiao et al., 2025), Tactic(Zhu et al., 2026), Evolving Sparsity(Han et al., 2026), and Louver(Dehghankar & Asudeh, 2026). InfLLM-V2(Zhao et al., 2026b) connects these stages with switchable dense-sparse attention.

6 Conclusion

In this paper, we introduced QASparse, a query-adaptive sparse attention method for efficient long-context prefilling in LLMs. QASparse combines Local Budget Regression and Boundary Voting to adapt computation budgets and retained KV tokens to different query regions and attention heads while reusing shared token rankings. Length-Specialized Kernels further improve execution efficiency by tailoring the computation to short and long contexts. Experiments across four model families on RULER and InfiniteBench show that QASparse achieves strong model performance relative to existing sparse attention methods, while latency measurements demonstrate favorable accuracy–efficiency trade-offs. These results highlight the value of combining query-adaptive token selection with efficient kernel design for long-context inference. Future work could extend this design to other hardware platforms and further reduce the overhead of adaptive token selection.

AI use statement

We used OpenAI Codex to assist with research ideation, method implementation, experiment design and execution, code debugging, result analysis, and manuscript writing. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final work.

References

Saurabh Agarwal, Bilge Acun, Basil Hosmer, Mostafa Elhoushi, Yejin Lee, Shivaram Venkataraman, Dimitris Papailiopoulos, and Carole-Jean Wu. CHAI: Clustered Head Attention for Efficient LLM Inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 291–312, 2024.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Guanzhong Chen. VSPrefill: Vertical-Slash Sparse Attention with Lightweight Indexing for Long-Context Prefilling, 2026. URL https://arxiv.org/abs/2603.04460.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models. In *International Conference on Learning Representations*, pp. 8220–8238, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/211ab571cc9f3802afa6ffff52ae3e5b-Abstract-Conference.html.

Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, May 2024.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-1189.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, et al. DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models, 2025. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, et al. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Mohsen Dehghankar and Abolfazl Asudeh. Sparse Attention as a Range Searching Problem: Towards an Inference-Efficient Index for KV Cache, 2026. URL https://arxiv.org/abs/2605.06763.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM Context Window Beyond 2 Million Tokens. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11091–11104. PMLR, July 2024.

Qihang Fan, Huaibo Huang, Zhiying Wu, Bingning Wang, and Ran He. FlashPrefill V2: Block-Sparse Prefill Attention for Long-Context LLM Serving, 2026a. URL https://arxiv.org/abs/2608.19758.

Qihang Fan, Huaibo Huang, Zhiying Wu, Juqiu Wang, Bingning Wang, and Ran He. FlashPrefill: Instantaneous Pattern Discovery and Thresholding for Ultra-Fast Long-Context Prefilling, March 2026b.

Yao Fu. Challenges in Deploying Long-Context Transformers: A Theoretical Peak Performance Analysis, May 2024.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data Engineering for Scaling Language Models to 128K Context. In *International Conference on Machine Learning*, volume 235, pp. 14125–14134. PMLR, 2024. URL https://proceedings.mlr.press/v235/fu24d.html.

Yoav Gelberg, Koshi Eguchi, Takuya Akiba, and Edoardo Cetin. Extending the Context of Pretrained LLMs by Dropping Their Positional Embedding. In *International Conference on Learning Representations*, volume 2026, pp. 60170–60197, April 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, et al. The Llama 3 Herd of Models, July 2024.

Ruizi Han, Miao Zhang, Ziyue Qiao, and Liqiang Nie. Evolving Sparsity: Leveraging Token Importance Dynamics for Efficient LLM Decoding with Sparse Attention. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11554–11566, 2026. doi: 10.18653/v1/2026.acl-long.530. URL https://aclanthology.org/2026.acl-long.530/.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the Real Context Size of Your Long-Context Language Models?, August 2024.

Yu Huo, Kun Zeng, Siyu Zhang, Yuquan Lu, Cheng Yang, Yifu Guo, and Xiaoying Tang. RepoShapley: Shapley-Enhanced Context Filtering for Repository-Level Code Completion. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 10390–10412, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.505.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating Pre-filling for Long-Context LLMs via Dynamic Sparse Attention. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. doi: 10.52202/079017-1663.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024b. doi: 10.18653/v1/2024.acl-long.91. URL https://aclanthology.org/2024.acl-long.91/.

Dongwon Jo, Beomseok Kang, Jiwon Song, and Jae-Joon Kim. Token Sparse Attention: Efficient Long-Context Inference with Interleaved Token Selection. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026.

Dalton Jones, Junyoung Park, Matthew Morse, Mingu Lee, Matthew Harper Langston, and Christopher Lott. QUOKA: Query-Oriented KV Selection For Efficient LLM Prefill. In *The Fourteenth International Conference on Learning Representations*, Rio de Janeiro, Brazil, April 2026.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. FlexPrefill: A Context-Aware Sparse Attention Mechanism for Efficient Long-Sequence Inference, February 2025.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM Knows What You are Looking for Before Generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970, 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/28ab418242603e0f7323e54185d19bde-Abstract-Conference.html.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, Yuanxing Zhang, Zhuo Chen, Hangyu Guo, Shilong Li, Ziqiang Liu, Yong Shan, Yifan Song, Jiayi Tian, Wenhao Wu, Zhejian Zhou, et al. A Comprehensive Survey on Long Context Language Modeling, November 2025.

Siran Liu, Zheng Cao, and Yongchao He. A Unified Sparse Attention via Multi-Granularity Compression. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a.

Siran Liu, Guoxia Wang, Sa Wang, Jinle Zeng, Haoyang Xie, Siyu Lou, Jiabin Yang, Dianhai Yu, Haifeng Wang, and Chao Yang. RRAtention: Dynamic Block Sparse Attention via Per-Head Round-Robin Shifts for Long-Context Inference. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26107–26124, San Diego, California, United States, July 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1199.

Amirkeivan Mohtashami and Martin Jaggi. Landmark Attention: Random-Access Infinite Context Length for Transformers, 2023. URL https://arxiv.org/abs/2305.16300.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave No Context Behind: Efficient Infinite Context Transformers with Infini-attention, 2024. URL https://arxiv.org/abs/2404.07143.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, et al. PyTorch: An imperative style, high-

performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient Context Window Extension of Large Language Models. In *The Twelfth International Conference on Learning Representations*, 2024.

Dan Peng, Zhihui Fu, Zewen Ye, Zhuoran Song, and Jun Wang. Accelerating Prefilling for Long-Context LLMs via Sparse Pattern Sharing, May 2025.

Ning Shang, Li Lyna Zhang, Siyuan Wang, Gaokai Zhang, Gilsinia Lopez, Fan Yang, Weizhu Chen, and Mao Yang. LongRoPE2: Near-Lossless LLM Context Window Scaling. In *International Conference on Machine Learning*, volume 267, pp. 54203–54218. PMLR, 2025. URL https://proceedings.mlr.press/v267/shang25a.html.

Jiwon Song, Dongwon Jo, Beomseok Kang, and Jae-Joon Kim. CompactAttention: Accelerating Chunked Prefill with Block-Union KV Selection, May 2026.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-Aware Sparsity for Efficient Long-Context LLM Inference. In *International Conference on Machine Learning*, volume 235, pp. 47901–47911. PMLR, 2024. URL https://proceedings.mlr.press/v235/tang24l.html.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Jingyu Sun, et al. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools, June 2024.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2019. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused Transformer: Contrastive Training for Context Scaling. In *Advances in Neural Information Processing Systems*, volume 36, pp. 42661–42688, 2023. doi: 10.52202/075280-1849. URL https://papers.neurips.cc/paper_files/paper/2023/hash/8511d06d5590f4bda24d42087802cc81-Abstract-Conference.html.

Xinghao Wang, Pengyu Wang, Dong Zhang, Chenkun Tan, Shaojun Zhou, Zhaoxiang Liu, Shiguo Lian, Fangxu Liu, Kai Song, and Xipeng Qiu. Sparser Block-Sparse Attention via Token Permutation. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026a.

Yixuan Wang, Huang He, Siqi Bao, Hua Wu, Haifeng Wang, Qingfu Zhu, and Wanxiang Che. ProxyAttn: Guided Sparse Attention via Representative Heads. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Chen Wu and Yin Song. Scaling Context, Not Parameters: Training a Compact 7B Language Model for Efficient Long-Context Processing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 61–68. Association for Computational Linguistics, July 2025. doi: 10.18653/v1/2025.acl-industry.6.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *International Conference on Learning Representations*, pp. 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5e5fd18f863cbe6d8ae392a93fd271c9-Abstract-Conference.html.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient Long-Context LLM Inference with Retrieval and Streaming Heads. In *International Conference on Learning Representations*, pp. 37228–37253,

2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5c1ddd2e59df46fd2aa85c833b1b36ed-Abstract-Conference.html.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, et al. Effective Long-Context Scaling of Foundation Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4643–4663, 2024. doi: 10.18653/v1/2024.naacl-long.260. URL https://aclanthology.org/2024.naacl-long.260/.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. Retrieval meets Long Context Large Language Models. In *International Conference on Learning Representations*, pp. 49569–49584, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/d75f29006df67df084e6586f1cb8458c-Abstract-Conference.html.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, et al. Qwen2.5-1M Technical Report, January 2025.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. FlashInfer: Efficient and Customizable Attention Engine for LLM Inference Serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/dbf02b21d77409a2db30e56866a8ab3a-Abstract-Conference.html.

Bozhi You, Irene Wang, Zelal Su Mustafaoglu, Abhinav Jangda, Angélica Moreira, Roshan Dathathri, Divya Mahajan, and Keshav Pingali. Flashlight: PyTorch Compiler Extensions to Accelerate Attention Variants. In *Proceedings of Machine Learning and Systems*, volume 8, pp. 874–891. MLSys, May 2026.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending Long Context Evaluation Beyond 100K Tokens, 2024a. URL https://arxiv.org/abs/2402.13718.

Yu Zhang, Songwei Liu, Chenqian Yan, Linsheng, Beichen Ning, Fangmin Chen, and Xing Wang. S2O: Early Stopping for Sparse Attention via Online Permutation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7737–7751. Association for Computational Linguistics, July 2026. doi: 10.18653/v1/2026.acl-long.351.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710, 2023. doi:

10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract.html.

Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu, and Zhangyang Wang. Found in the Middle: How Language Models Use Long Contexts Better via Plug-and-Play Positional Encoding. In *Advances in Neural Information Processing Systems*, volume 37, pp. 60755–60775, 2024b. doi: 10.52202/079017-1943. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6ffdbbe354893979367f93e2121e37dd-Abstract-Conference.html.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, Xiaolei Wang, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, et al. A Survey of Large Language Models. *Frontiers of Computer Science*, 20(12):2012627, May 2026a. doi: 10.1007/s11704-026-60308-3.

Weilin Zhao, Zihan Zhou, Zhou su, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, Xu Han, and Zhiyuan Liu. InfLLM-V2: Dense-Sparse Switchable Attention for Seamless Short-to-Long Adaptation. In *International Conference on Learning Representations*, pp. 69374–69390, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/716e60181e1782f5bd0a629033f85310-Abstract-Conference.html.

Kan Zhu, Tian Tang, Qinyu Xu, Zhan Jin, Yile Gu, Zhichen Zeng, Rohan Kadekodi, Liangyu Zhao, Ang Li, Arvind Krishnamurthy, and Baris Kasikci. Tactic: Adaptive Sparse Attention with Clustering and Distribution Fitting for Long-Context LLMs. In *International Conference on Learning Representations*, pp. 30626–30644, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/33f94d79acf71051d6a27f4d8889e20e-Abstract-Conference.html.

A Attention Ordering and Generation Quality

We compare two token-selection orders during decoding using Llama-3.1-8B base on RULER at 4k. At each decoding step, we compute full attention and retain tokens in descending (*high-to-low*) or ascending (*low-to-high*) order of attention weight until their cumulative mass reaches 20%, 30%, or 40%. All settings use the same 20 fixed examples from each of the 13 RULER tasks and full attention during prefill.

Table 3: RULER scores of Llama-3.1-8B base at 4k, averaged equally over 13 tasks with 20 fixed examples per task.

Threshold	High-to-low	Low-to-high	Gap
20%	67.53	19.19	+48.33
30%	77.79	45.69	+32.10
40%	84.92	70.15	+14.77

Prioritizing high-attention tokens gives better generation quality at every tested coverage threshold, supporting attention weight as a criterion for token selection under a limited budget.

B Number of Shared Rankings

Tables 4–7 report the RULER scores of ProxyAttn and QASparse on four models with different numbers of shared rankings. Each model is evaluated at six context lengths from 4k to 128k; Mean summarizes these lengths. Within each ranking-count pair, the higher score in each column is bold; ties highlight QASparse.

Table 4: RULER scores of GLM-4-9B-Chat-1M across shared-ranking counts.

Method	Rankings	4k	8k	16k	32k	64k	128k	Mean ↑
ProxyAttn	1	95.00	**93.67**	91.73	**90.57**	**87.76**	85.45	90.70
QASparse	1	**95.05**	93.46	**92.97**	90.39	87.53	**86.03**	**90.91**
ProxyAttn	2	**95.31**	93.25	92.21	90.21	**87.60**	85.97	90.76
QASparse	2	95.29	**93.77**	**93.31**	**91.24**	86.05	**86.31**	**90.99**
ProxyAttn	4	95.21	93.73	92.44	**90.62**	**87.70**	85.61	90.89
QASparse	4	**95.36**	**93.85**	**93.33**	90.22	86.86	**86.01**	**90.94**

Table 5: RULER scores of Llama-3.1-8B-Instruct across shared-ranking counts.

Method	Rankings	4k	8k	16k	32k	64k	128k	Mean ↑
ProxyAttn	1	94.12	92.25	93.35	90.34	87.24	75.15	88.74
QASparse	1	**94.90**	**93.21**	**93.75**	**92.22**	**87.70**	**77.82**	**89.93**
ProxyAttn	2	**94.68**	**92.55**	**93.39**	90.17	86.52	**77.60**	**89.15**
QASparse	2	94.29	91.91	93.34	**90.56**	**87.11**	77.40	89.10
ProxyAttn	4	94.42	**93.05**	94.03	90.61	86.00	**78.04**	89.36
QASparse	4	**94.82**	92.73	**94.28**	**91.73**	**86.48**	77.35	**89.56**
ProxyAttn	8	**95.28**	**93.21**	**94.39**	90.71	86.41	76.68	89.44
QASparse	8	95.22	92.83	93.94	**91.25**	**87.12**	**77.17**	**89.59**

Table 6: RULER scores of MegaBeam-Mistral-7B-512k across shared-ranking counts.

Method	Rankings	4k	8k	16k	32k	64k	128k	Mean ↑
ProxyAttn	1	93.51	90.84	**88.43**	**87.09**	**83.80**	84.39	88.01
QASparse	1	**94.21**	**92.30**	87.96	86.94	83.00	**85.30**	**88.29**
ProxyAttn	2	93.13	90.32	**86.74**	**86.72**	**83.98**	84.82	87.62
QASparse	2	**93.80**	**91.03**	86.63	86.12	82.76	**85.62**	**87.66**
ProxyAttn	4	93.71	90.92	**87.23**	**87.49**	**84.22**	**84.63**	88.03
QASparse	4	**94.09**	**91.76**	87.02	87.25	83.42	84.62	**88.03**
ProxyAttn	8	**93.67**	91.53	88.11	87.17	**84.13**	85.44	88.34
QASparse	8	93.63	**92.23**	**88.49**	**87.57**	83.76	**85.56**	**88.54**

Table 7: RULER scores of Qwen2.5-7B-Instruct-1M across shared-ranking counts.

Method	Rankings	4k	8k	16k	32k	64k	128k	Mean ↑
ProxyAttn	1	94.17	91.91	92.42	87.60	86.04	81.17	88.88
QASparse	1	**94.35**	**92.21**	**92.91**	**88.95**	**86.54**	**82.52**	**89.58**
ProxyAttn	2	**94.63**	91.58	92.30	87.74	86.82	81.27	89.06
QASparse	2	94.24	**92.23**	**93.14**	**88.76**	**87.92**	**83.08**	**89.89**
ProxyAttn	4	94.29	**91.71**	92.63	88.80	86.93	82.27	89.44
QASparse	4	**94.51**	91.54	**92.87**	**89.31**	**87.84**	**83.73**	**89.97**

C Boundary Voting Candidate-Block Count

Table 8 reports the candidate-block sweep discussed in Section 4.3, using Llama-3.1-8B-Instruct on RULER. The default 16-block setting takes up to eight blocks on each side of the shared-ranking cutoff, with a block traversal stride of one. Query and key vectors are sampled every eight positions within each 128-token block. These counts refer to total candidate *blocks*, whereas r in Section 3 denotes candidate *tokens* on each side. Scores are compared within each table; the component ablations and shared-ranking sweep use separate evaluation runs.

Table 8: RULER scores of Llama-3.1-8B-Instruct for different total candidate-block counts.

Setting	4k	8k	16k	32k	64k	128k	Mean ↑
4 candidate blocks	92.67	92.29	91.69	88.90	88.73	80.41	89.12
8 candidate blocks	92.74	92.55	91.50	89.13	89.00	79.55	89.08
16 candidate blocks	92.67	93.51	91.76	89.67	88.92	78.71	89.21
32 candidate blocks	91.78	92.74	91.83	89.44	88.40	79.68	88.98

D Short- and Long-Context Kernel Timing

Table 9 gives the selection-kernel latencies underlying Figure 5, for seven context lengths from 8k to 512k. For short- and long-context latencies $t_s(L)$ and $t_l(L)$, the reference is their geometric mean, and each speedup is the reference divided by the corresponding latency:

$$t_{\mathrm{ref}}(L) = \sqrt{t_s(L)t_l(L)}, \qquad S_s(L) = \frac{t_{\mathrm{ref}}(L)}{t_s(L)}, \qquad S_l(L) = \frac{t_{\mathrm{ref}}(L)}{t_l(L)}.$$

Table 9: Selection-kernel latency and speedup relative to the geometric-mean latency at each context length.

Length	Short (ms)	Long (ms)	Reference (ms)	Short speedup (×)	Long speedup (×)
8k	0.42	0.77	0.569	1.354	0.739
16k	1.16	1.50	1.319	1.137	0.879
32k	4.10	2.56	3.240	0.790	1.266
64k	78.28	5.31	20.388	0.260	3.840
128k	460.30	11.82	73.761	0.160	6.240
256k	1165.54	35.51	203.441	0.175	5.729
512k	6848.10	148.62	1008.843	0.147	6.788

E Local Budget Regression Scheme

Table 10 compares the three regression schemes defined in Section 3 on Llama-3.1-8B-Instruct across the six RULER lengths. All schemes use one representative query per 128-token query block. Non-overlapping Piecewise Linear Regression uses 10 sampled queries per region. Sliding-window Linear Regression uses windows of 10 sampled queries, a stride of two sampled queries, and averages overlapping predictions.

Table 10: RULER scores of Llama-3.1-8B-Instruct for the three budget-regression schemes.

Setting	4k	8k	16k	32k	64k	128k	Mean ↑
Global Linear Regression	92.74	92.74	91.83	88.51	89.15	76.74	88.62
Non-overlapping Piecewise Linear Regression	92.74	92.74	91.06	89.74	89.04	79.68	89.17
Sliding-window Linear Regression	92.67	93.51	91.76	89.67	88.92	78.71	89.21

F Full Attention on the Final Two Query Blocks

This experiment ablates the final-mask rule that gives full attention to the last two query blocks. In the default configuration, these blocks attend to all causally visible key blocks, overriding the sparse selection for their query positions. The control removes only this override: the final two query blocks instead use the sparse mask determined by the shared ranking, Local Budget Regression, and Boundary Voting. The remaining settings are unchanged.

We compare the two configurations on Llama-3.1-8B-Instruct over the six RULER context lengths. This isolates whether full attention at the end of the prefill sequence contributes to the quality of the resulting generation.

Table 11: RULER scores with and without the final-two-query-block full-attention override.

Setting	4k	8k	16k	32k	64k	128k	Mean ↑
Force final two query blocks	92.67	93.51	91.76	89.67	88.92	78.71	89.21
No final-block override	91.90	90.21	86.08	83.40	83.65	72.88	84.69

Retaining full attention for the final two query blocks improves scores at every tested length. The consistent advantage supports keeping this rule in the final attention mask.

G Ranking Used for Budget-Label Computation

This experiment changes the ordering used to construct the sampled budget labels in Local Budget Regression. Both variants compute the target head's actual post-RoPE attention mass and accumulate it toward the same coverage threshold. The default traverses key blocks in the proxy/shared

ranking; the control traverses them in descending order of their true attention mass. Both variants use the shared ranking for initial selection, followed by the same Boundary Voting procedure, so only the ordering used to determine the budget labels is changed.

For normalized attention mass a_j over causally visible key blocks, the label under ordering π is

$$b(\pi) = \min \left\{ m : \sum_{j=1}^{m} a_{\pi(j)} \geq \gamma \right\}.$$

Sorting by true mass reaches the threshold with no more blocks than the shared ordering. However, applying that budget to blocks selected in shared order may retain less mass than the label assumes. The comparison therefore tests whether budget estimation should follow the ordering used for selection. Table 12 reports both variants on Llama-3.1-8B-Instruct over the six RULER lengths, with the remaining settings unchanged.

Table 12: RULER scores for shared and true orderings in budget-label computation; initial selection uses the shared ranking in both cases.

Setting	4k	8k	16k	32k	64k	128k	Mean ↑
Shared ranking for budget label	92.67	93.51	91.76	89.67	88.92	78.71	89.21
True ranking for budget label	91.90	92.63	90.83	89.17	86.85	76.46	87.97

Using shared-order labels gives higher scores at every tested length, supporting alignment between budget estimation and the ordering used for selection. The experiment compares budget-label construction, rather than replacing the selection ranking with the true ranking.

H Max Voting versus Exponential-Sum Voting

This experiment ablates the candidate-block scoring rule in Boundary Voting. Let K_j contain the sampled key vectors of candidate block j, and let $\ell(q, k)$ denote the post-RoPE QK logit for sampled query q and key k. Max voting scores a block by its strongest sampled interaction, whereas exponential-sum voting accumulates the exponentiated logits within the block:

$$s_j^{\max}(q) = \max_{k \in K_j} \ell(q, k), \qquad s_j^{\exp}(q) = \sum_{k \in K_j} \exp\big(\ell(q, k)\big).$$

The exponential sum is computed with numerical stabilization, and only causally visible pairs are scored. In both variants, each sampled query casts exactly one vote for the candidate block with the highest score. Candidate blocks are then ranked by their vote counts, and the same number of blocks is retained. The candidate region, query/key sampling, and selected-block count are unchanged, so the ablation compares the evidence used to cast each vote rather than changing the voting budget.

Table 13 compares the two scoring rules on Llama-3.1-8B-Instruct across the six RULER lengths.

Table 13: RULER scores for maximum and exponential-sum scoring in Boundary Voting.

Setting	4k	8k	16k	32k	64k	128k	Mean ↑
Max voting	92.67	93.51	91.76	89.67	88.92	78.71	89.21
Exponential-sum voting	91.97	92.74	91.83	89.67	89.46	78.91	89.10

Max voting achieves the higher mean RULER score, supporting its use as the default voting rule.

I PARETO TRADE-OFF MEASUREMENTS

We evaluate Llama-3.1-8B-Instruct on RULER while varying the attention-mass threshold γ for ProxyAttn and QASparse. Tables 14 and 15 report the score and attention latency of every measured configuration at six context lengths. Avg retains the aggregate reported with each measurement. Figure 4 plots these paired measurements for the average, 4k, and 64k settings. Its Full Attention reference scores are 89.10, 95.18, and 85.56, respectively, from the Llama RULER results in Table 1.

Table 14: RULER scores of Llama-3.1-8B-Instruct across attention-mass thresholds and context lengths.

Method	γ	**4k**	**8k**	**16k**	**32k**	**64k**	**128k**	**Avg**
ProxyAttn	0.97	93.59	93.59	95.77	92.90	85.18	74.42	89.24
	0.95	93.59	92.05	94.92	91.10	84.33	73.68	88.28
	0.85	92.82	89.55	91.14	86.82	83.37	67.79	85.25
	0.80	91.86	84.42	85.72	82.86	80.29	65.45	81.77
	0.75	91.09	81.65	84.03	81.24	79.29	62.50	79.97
	0.70	88.40	83.65	81.95	80.92	76.26	59.83	78.50
	0.65	88.72	80.06	84.91	80.35	74.76	57.42	77.70
	0.60	85.90	75.68	81.99	79.46	74.65	60.10	76.30
QASparse	0.97	94.36	94.62	95.91	92.94	86.81	76.31	90.16
	0.95	93.59	94.04	94.69	91.86	88.45	75.54	89.69
	0.85	93.21	92.55	94.15	91.85	85.45	72.53	88.29
	0.80	93.21	93.38	92.14	89.88	83.91	69.60	87.02
	0.75	92.24	91.85	90.12	88.42	82.51	67.47	85.44
	0.70	91.36	90.01	89.64	82.81	80.15	64.35	83.05
	0.65	90.44	90.96	85.03	81.35	78.72	61.27	81.29
	0.60	89.78	88.65	83.82	82.08	76.40	58.74	79.91

Table 15: Attention latencies (ms) corresponding to the RULER scores in Table 14.

Method	γ	**4k**	**8k**	**16k**	**32k**	**64k**	**128k**	**Avg**
ProxyAttn	0.97	2.18	3.89	9.47	26.66	84.97	289.73	69.48
	0.95	2.98	4.12	8.23	21.69	67.18	224.11	54.72
	0.85	1.98	3.25	6.55	13.83	37.42	112.54	29.26
	0.80	2.11	3.16	6.14	12.16	30.90	91.79	24.38
	0.75	1.96	2.96	5.09	10.60	27.11	78.09	20.97
	0.70	1.96	3.12	5.17	9.99	24.37	67.88	18.75
	0.65	1.93	2.80	4.66	9.39	22.31	60.03	16.85
	0.60	2.07	2.75	4.50	9.27	21.36	53.93	15.65
QASparse	0.97	0.89	2.56	8.06	25.08	82.78	363.04	80.40
	0.95	0.84	2.31	6.97	21.41	67.24	300.62	66.56
	0.85	0.72	1.78	4.88	13.83	39.92	164.21	37.56
	0.80	0.70	1.69	4.47	12.30	35.01	134.35	31.42
	0.75	0.68	1.61	4.17	11.48	31.09	116.07	27.52
	0.70	0.66	1.54	3.91	10.50	28.32	100.53	24.24
	0.65	0.65	1.49	3.68	9.75	25.65	87.13	21.39
	0.60	0.64	1.46	3.56	9.10	23.71	76.55	19.17

J ATTENTION LATENCY MEASUREMENTS

Table 16 reports a separate attention-latency measurement of Llama-3.1-8B-Instruct at six context lengths, comparing FlashAttention with QASparse at $\gamma = 0.90$.

Table 16: Attention latencies (ms) of Llama-3.1-8B-Instruct.

Method	**4k**	**8k**	**16k**	**32k**	**64k**	**128k**
FlashAttention	0.71	2.48	9.47	36.57	143.77	595.57
QASparse ($\gamma = 0.90$)	0.77	1.97	5.61	16.44	48.61	211.74